Titulación: Grado en Ingeniería Informática

Asignatura: Fundamentos de Computadores

Bloque 1: Introducción

Tema 2: Sistema binario de representación numérica

Pablo Huerta Pellitero

ÍNDICE

BIBLIOGRAFÍA

- Román Hermida, Ana Mº del Corral, Enric Pastor, Fermín Sánchez
  "Fundamentos de Computadores" , cap 1
  Editorial Síntesis
- Thomas L. Floyd
  "Fundamentos de Sistemas Digitales", cap 2
  Editorial Prentice Hall
- Daniel D. Gajski
  "Principios de Diseño Digital", cap 2
  Editorial Prentice Hall
- M. Morris Mano
  "Diseño Digital", cap 1
  Editorial Prentice Hall

ÍNDICE

SISTEMAS DE NUMERACIÓN DE BASE FIJA

- Un sistema numérico consta de:
  - Un conjunto ordenado de símbolos (cifras o dígitos).
  - Relaciones definidas para la suma, resta, multiplicación y división.
- Se denomina **base** de un sistema numérico al número de cifras o dígitos que utiliza.
- Ejemplos:
  - Sistema decimal (base = 10).
    - Dígitos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Sistema binario (base = 2).
    - Dígitos: {0, 1}
  - Sistema hexadecimal (base = 16).
    - Dígitos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
  - Sistema octal (base = 8).
    - Dígitos: {0, 1, 2, 3, 4, 5, 6, 7}

SISTEMAS DE NUMERACIÓN DE BASE FIJA

- Los números en un sistema de numeración consisten en una secuencia de dígitos que pueden tener parte entera y parte fraccionaria separadas por una coma:

$$N = (a_{p-1}\ a_{p-2}\ \ldots\ a_1\ a_0\ ,\ a_{-1}\ a_{-2}\ \ldots\ a_{-q})_r$$

  a_i son los dígitos,
  p es el número de dígitos enteros,
  q es el número de dígitos fraccionarios,
  a_{p-1} es el dígito más significativo,
  a_{-q} es el dígito menos significativo.

- Ejemplo:
  - $(107,45)_{10}$
  - $(1A3,B)_{16}$
  - $(101110,11)_2$
  - $(1473,13)_8$
- Esta forma de representar un número se conoce como **notación posicional**.

SISTEMAS DE NUMERACIÓN DE BASE FIJA

- Se define el peso del dígito a_i como r^i .
- Cada dígito tiene asociado un valor en función de su peso, y es el producto de dicho dígito por el peso.
  - Valor de a_i : $a_i \cdot r^i$
- Un número se puede representar como la suma de los valores de todos sus dígitos:

$$N = \sum_{i=-q}^{p} a_i \cdot r^i$$

- Ejemplo: el número en notación posicional $(123,54)_{10}$ se puede representar como :

$$(123,54)_{10} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 5 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

- Esta forma de representar un número se conoce como **notación polinomial**.

ÍNDICE

CONVERSIÓN ENTRE BASES

- Dos métodos diferentes: sustitución en serie y división/multiplicación por la base.
- Sustitución en serie: se utiliza para pasar un número en base *r* a base *s*, utilizando las operaciones de la base *s*.
  - Sólo hay que evaluar la notación polinomial del número utilizando las operaciones de la base *s:*
  - Ejemplo: convertir el número $(101,01)_2$ a decimal.

$$(101,01)_2 = 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} =$$
$$= 4 + 0 + 1 + 0 + 0,25 = (5,25)_{10}$$

  - Ejemplo: convertir el número $(1AC,2)_{16}$ a decimal.

$$(1AC,2)_{16} = 1\cdot 16^2 + A\cdot 16^1 + C\cdot 16^0 + 2\cdot 16^{-1} =$$
$$= 1\cdot 256 + 10\cdot 16 + 12\cdot 1 + 2\cdot 0,0625 = (428,125)_{10}$$

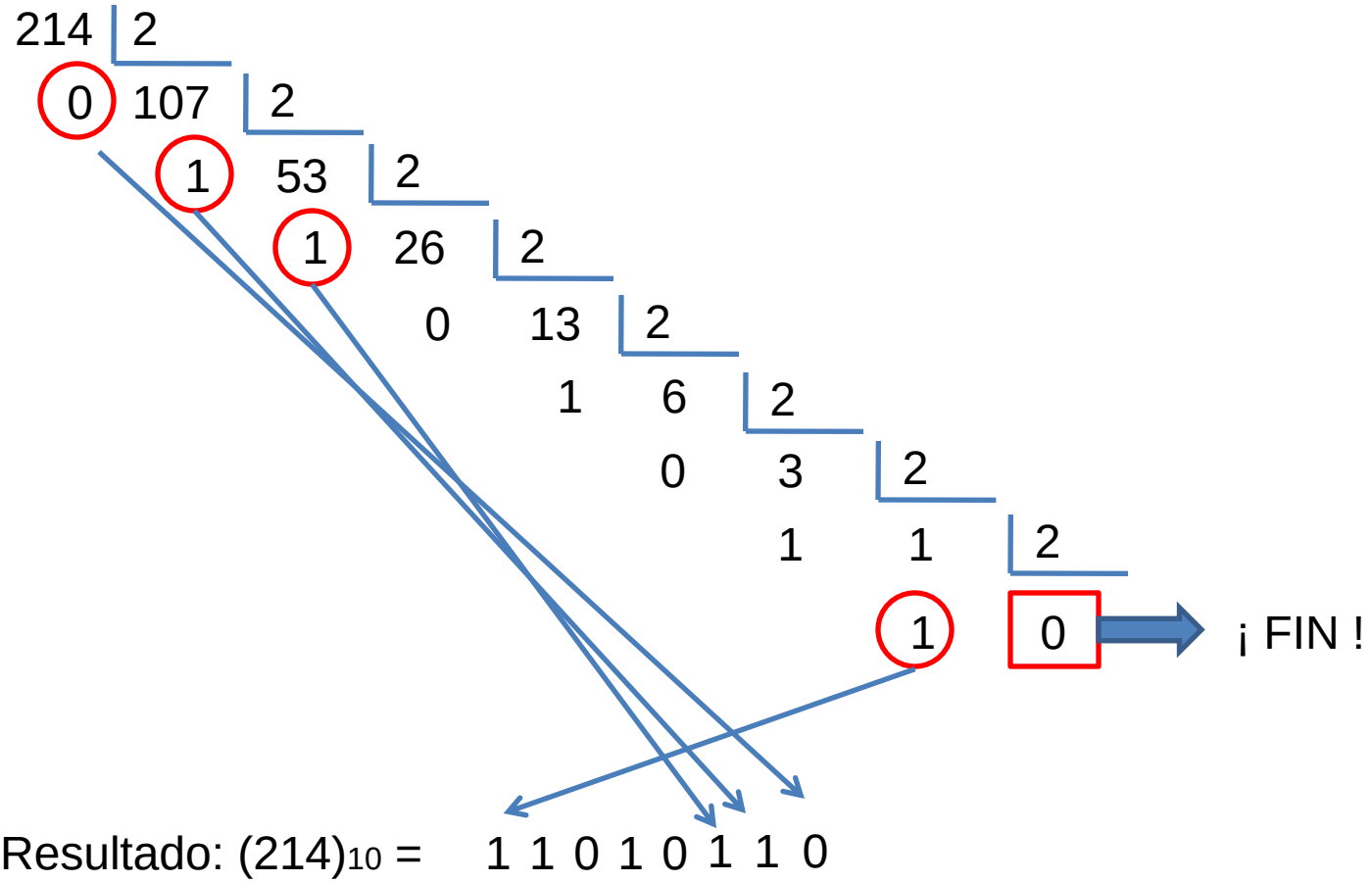
CONVERSIÓN ENTRE BASES

- División/multiplicación por la base: se utiliza para pasar un número en base *r* a base *s* utilizando las operaciones de la base *r.*

  - Para la parte entera:
    - Se divide la parte entera entre la base *s*. El resto obtenido será el dígito a_0 de la parte entera.
    - El cociente se divide de nuevo entre la base *s*, obteniéndose un nuevo resto que será a_1 .
    - Se continúa realizando divisiones sucesivas obteniendo nuevos dígitos, hasta que el cociente vale '0'.
  - Para la parte fraccionaria:
    - Se multiplica la parte fraccionaria por la base *s*, obteniéndose un nuevo número que tendrá parte entera y parte fraccionaria. La parte entera será el dígito a_{-1} .
    - La nueva fraccionaria se vuelve a multiplicar por la base, obteniendo un nuevo dígito a_{-2} .
    - Se continúa hasta que se obtiene una parte fraccionaria igual a '0', ó se tienen suficientes dígitos.

CONVERSIÓN ENTRE BASES

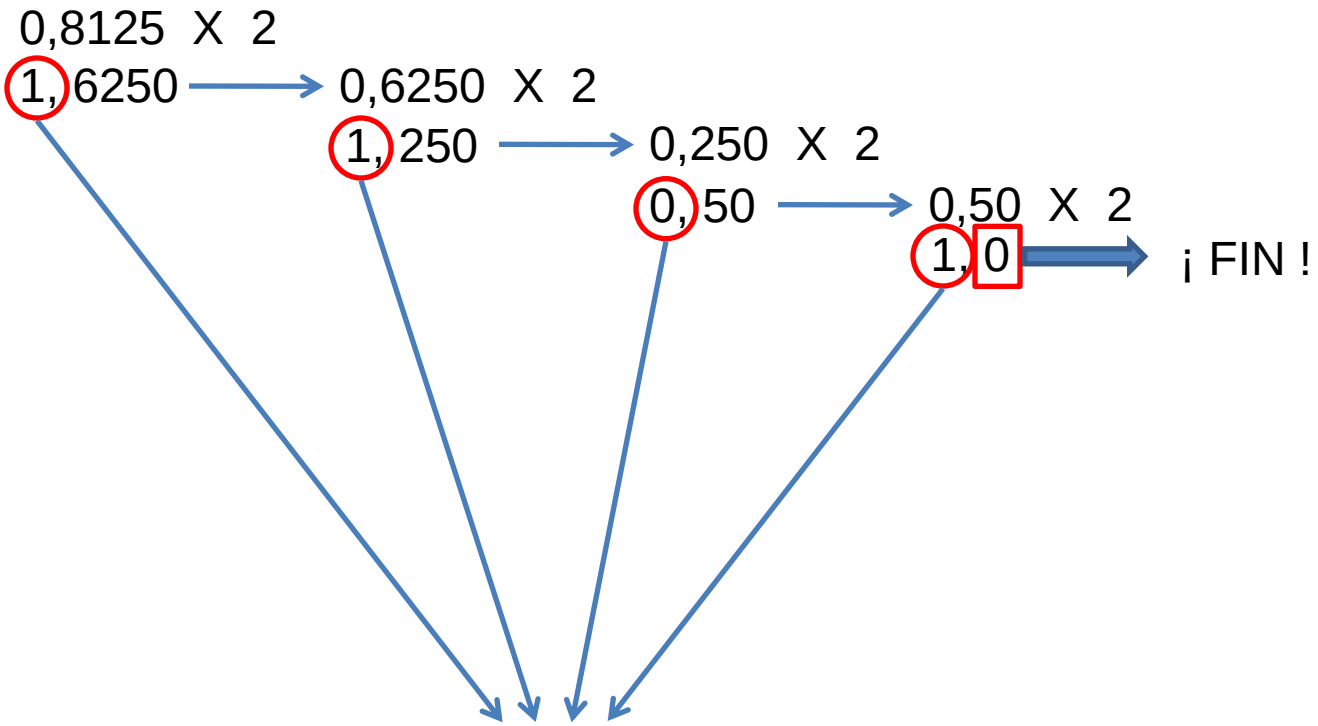
- Ejemplo: convertir el número $(214)_{10}$ a binario.



Resultado: $(214)_{10}$ = 1 1 0 1 0 1 1 0

CONVERSIÓN ENTRE BASES

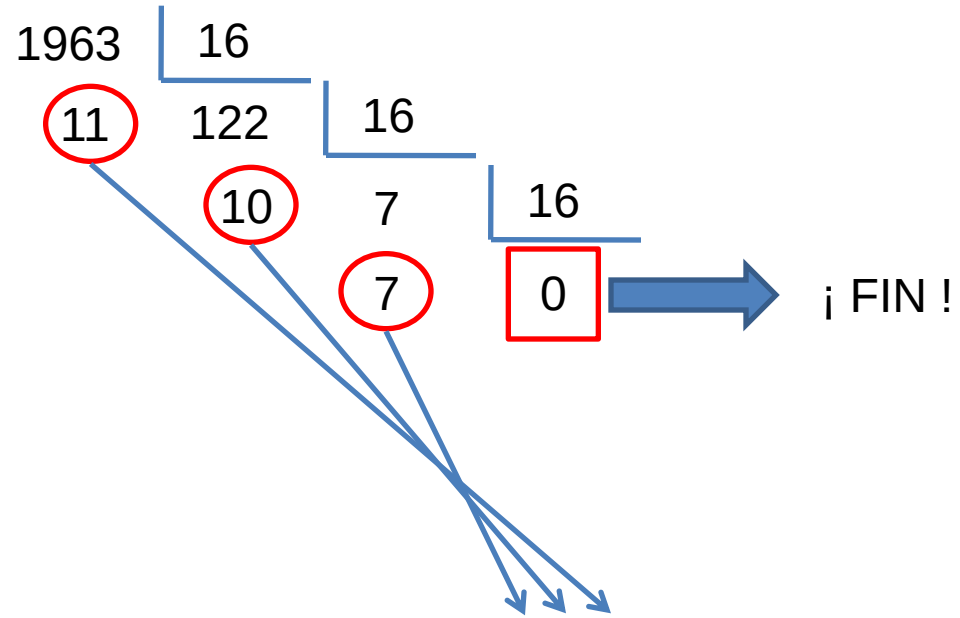
- Ejemplo: convertir el número $(0,8125)_{10}$ a binario.



Resultado: $(0,8125)_{10} = 0,1\,1\,0\,1$

CONVERSIÓN ENTRE BASES

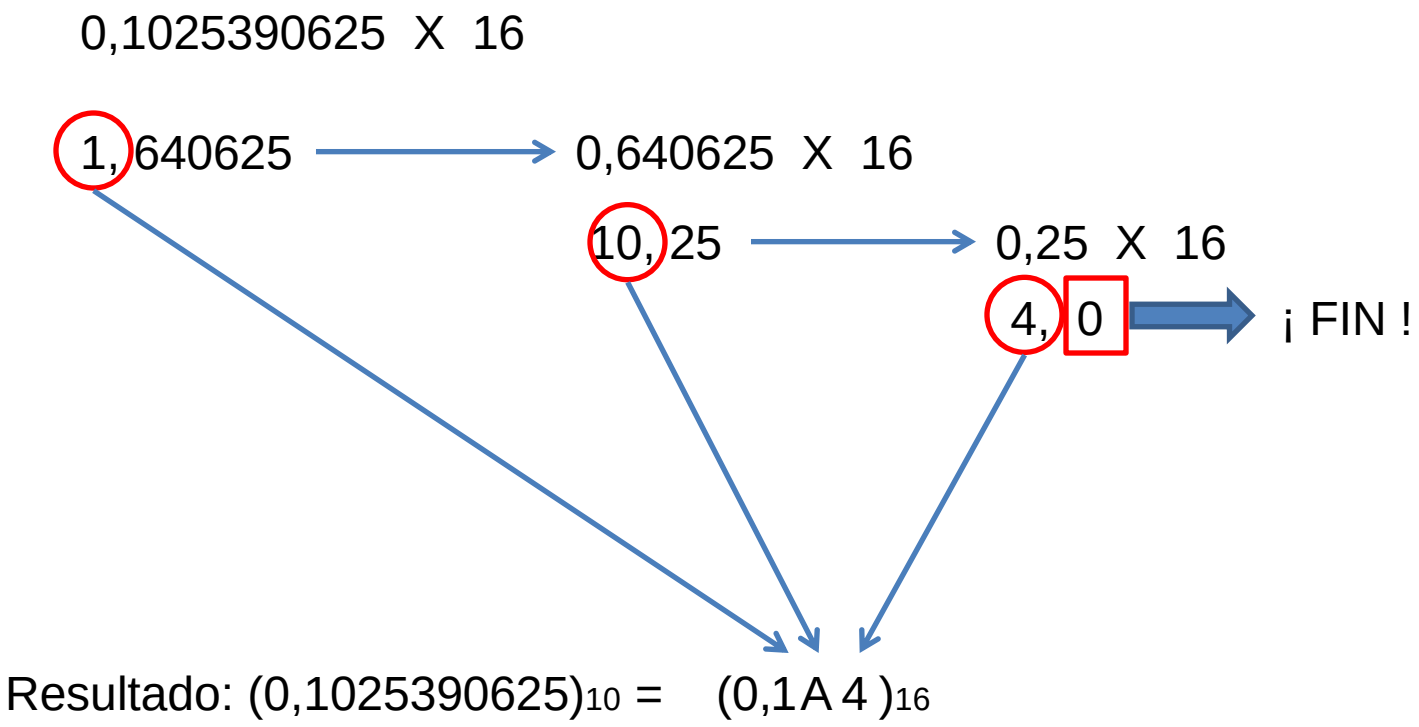
- Ejemplo: convertir el número $(1963)_{10}$ a hexadecimal.



Resultado: $(1963)_{10} = (7\ A\ B)_{16}$

CONVERSIÓN ENTRE BASES

- Ejemplo: convertir el número $(0,1025390625)_{10}$ a decimal.



Resultado: $(0,1025390625)_{10} = (0,1A4)_{16}$

CONVERSIÓN ENTRE BASES

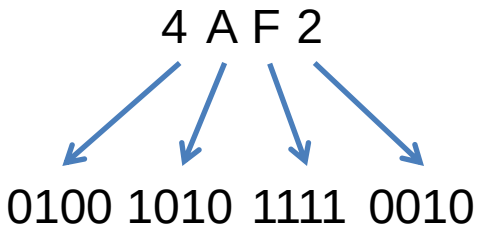
Correspondencias de los 15 primeros números en distintas bases

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binario	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

CONVERSIÓN ENTRE BASES

- Para convertir de binario a hexadecimal y viceversa, así como de binario a octal, existe un método sencillo.
- **Binario => hexadecimal:** se completa con '0' por la izquierda hasta que se tenga un número de bits múltiplo de 4. Se agrupan los bits de 4 en 4 y cada grupo de 4 bits se convierte directamente a hexadecimal:

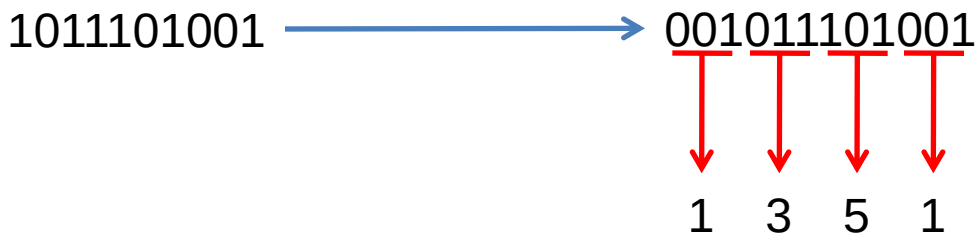
- **Hexadecimal => binario:** cada dígito hexadecimal se sustituye por su correspondiente valor en binario con 4 bits:



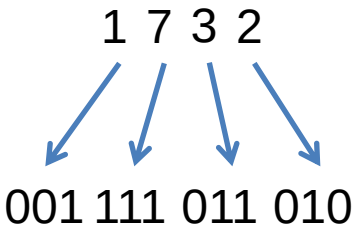
CONVERSIÓN ENTRE BASES

- Para pasar de binario a octal y viceversa el proceso es el mismo, pero agrupando los bits de 3 en 3:

Binario a octal



Octal a binario



ÍNDICE

ARITMÉTICA BINARIA

- El sistema binario es el utilizado por los computadores para la representación de los números por dos razones:
  - Los dispositivos electrónicos que manejan señales con sólo dos posibles valores son mas sencillos y baratos de fabricar.
  - Las operaciones aritméticas en binario son sencillas de implementar.
- En el sistema binario cualquier número se representa utilizando solamente los dígitos ‘0’ y ‘1’.
- Para referirse a los dígitos binarios se utiliza habitualmente el término **bit** (**bi**nary dig**it**).
- Las tablas de las operaciones aritméticas básicas son:

+	0	1
0	0	1
1	1	10

x	0	1
0	0	0
1	0	1

ARITMÉTICA BINARIA

- Ejemplos:

Sumar: 1001 + 1011

```
  1   1 1      acarreo
    1 0 0 1
+   1 0 1 1
-----------
  1 0 1 0 0
```

Sumar: 1011 + 0011

```
      1 1      acarreo
    1 0 1 1
+   0 0 1 1
-----------
    1 1 1 0
```

Restar: 101101 - 011011

```
    1 0 1 1 0 1
-   0 1 1 0 1 1
---------------
      1     1    acarreo
---------------
    0 1 0 0 1 0
```

Restar: 101101 - 011110

```
    1 0 1 1 0 1
-   0 1 1 1 1 0
---------------
      1 1 1 1    acarreo
---------------
    0 0 1 1 1 1
```

ARITMÉTICA BINARIA

- Ejemplos:

Multiplicar: 1100 X 101

```
    1 1 0 0
  x   1 0 1
-------------
    1 1 0 0
  0 0 0 0
1 1 0 0
-------------
1 1 1 1 0 0
```

Dividir: 1101001 / 101

```
1 1 0 1 0 0 1 | 1 0 1
1 0 1           -----
-----           1 0 1 0 1   → Cociente
  0 0 1 1
  0 0 0
  -----
    1 1 0
    1 0 1
    -----
    0 0 1 0
      0 0 0
      -----
        1 0 1
        1 0 1
        -----
        0 0 0   → Resto
```

ÍNDICE

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- En un computador los tamaños de los datos que se pueden manejar no son arbitrarios, sino que van ligados a la arquitectura del computador y a la circuitería utilizada.
- El computador va a manejar datos que tienen tamaños fijos: 8 bits, 16 bits, 32 bits, etc.
- Con un número fijo 'n' de bits se pueden representar 2^n valores o números diferentes, además de otro tipo de información como caracteres o imágenes.
- Vamos a estudiar diferentes formas de representar números enteros utilizando 'n' bits.
- Uno de los criterios más importantes a la hora de escoger una forma de representación u otra es que las operaciones aritméticas sean lo más sencillas posibles de implementar.
- Al ser el tamaño de los datos fijo, puede suceder que al realizar alguna operación aritmética el resultado no sea representable con el número de bits disponible => ***Overflow*** **o desbordamiento**.

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- Representación de enteros sin signo:
  - Con ‘n’ bits se pueden representar los números del 0 al 2^n-1 utilizando el sistema de numeración binario.
  - Suma: la suma de 2 números de ‘n’ bits puede necesitar ‘n+1’ bits para representar el resultado => *Overflow*.

```
  1 0 1 1
+ 0 1 1 1
---------
1 0 0 1 0
↑
overflow
```

  - Resta: la operación de resta puede dar un resultado < 0 => *Overflow*.
  - Multiplicación: el resultado de una multiplicación de 2 número de ‘n’ bits puede necesitar hasta ‘2n’ bits para ser representada => *Overflow*.
  - Las operaciones de suma y resta son diferentes. Sería interesante que una suma y una resta se realizasen de la misma forma => **utilizar complemento a la base para las restas**.

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- **Complemento a la base de un número**: para un número N de ‘n’ dígitos en base ‘r’, se define el complemento a la base de N como C_rN:

$$C_r N = r^n - N$$

- Para el caso binario la base es 2, y el complemento se llama “complemento a 2”. Por ejemplo para el número N = 1001 de 4 bits:

$$C_2 N = 2^4 - N$$

$$C_2(1001) = 10000 - 1001 = 0111$$

- El complemento a 2 permite realizar la operación de resta como una suma. Ejemplo: siendo A y B números enteros sin signo de ‘n’ bits:

$$A + C_2 B = A + 2^n - B = (A - B) + 2^n$$

- Tenemos por una parte 2^n, que tiene sus ‘n’ bits menos significativos igual a ‘0’, y por otra parte (A – B) que tiene ‘n’ bits. Luego el resultado de la resta de A y B se puede obtener haciendo la suma de A + C_2B.

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- Pero para calcular el C_2N es necesario hacer una resta 2^n-N, luego ¿de qué vale si seguimos teniendo que hacer restas?
- Se define el complemento a 1 de N (**complemento restringido a la base**) como:
$$C_1N = 2^n - N - 1 = C_2N - 1$$
- Es decir que:
$$C_2N = C_1N + 1$$
- Por tanto podemos calcular el complemento a 2 de un número utilizando una suma en lugar de una resta.
- ¿Pero cómo se calcula el complemento a 1 de un número N representado en binario?
- Muy sencillo, basta con cambiar los ‘1’ por ‘0’ y los ‘0’ por ‘1’.
- Ejemplo: $C_1(1011) = 0100$
- Conclusión: se pueden realizar restas utilizando sólo operaciones de suma, lo cual simplifica mucho los circuitos que realizan este tipo de operaciones

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- Representación de enteros con signo: signo y magnitud.
  - Utilizando ‘n’ bits podemos representar número enteros con signo, utilizando el bit más significativo como signo y los ‘n-1’ bits restantes como magnitud.
  - De esta forma, podemos representar números entre $2^{n-1}-1$ y $-(2^{n-1}-1)$.
  - Con esta representación hay dos codificaciones para el número ‘0’, una con signo positivo y otra con signo negativo.

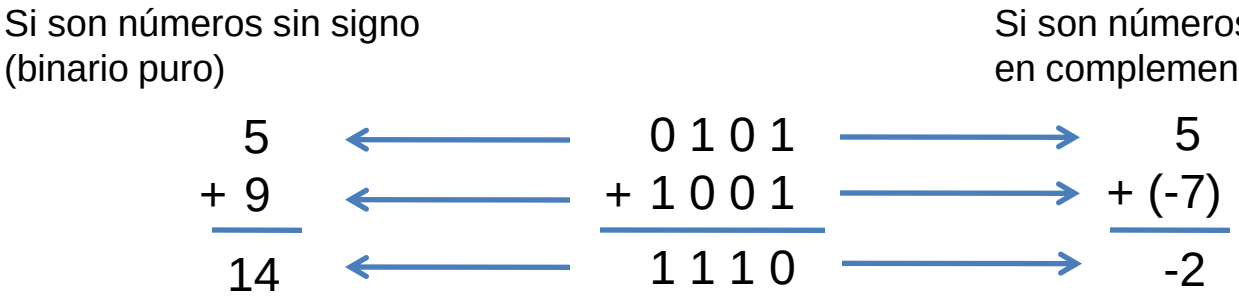
- Sumas y restas en signo y magnitud.
  - Hay que tener en cuenta los signos de los operandos.
  - Hay que ordenar los módulos si hay que restar.
  - Hay que calcular el signo del resultado en función de los signos de los operandos y sus módulos.
- Conclusión: hay que construir circuitos que tengan en cuenta todas las posibilidades de signos y módulos para poder realizar sumas y restas correctamente, por lo que no es la representación que se usa habitualmente en los computadores.

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- Representación de enteros con signo: complemento a 2.
  - Utilizando ‘n’ bits se pueden representar enteros con signo:
    - Los positivos se representan en binario natural.
    - Los negativos se representan como el C_2 del positivo correspondiente.
    - Para cambiar el signo de un número, se hace el C_2 de dicho número.
  - Ejemplo con ‘n’ = 4 bits:
    - 6_{10} => 0110
    - -6_{10} => $C_2(6_{10}) = C_2(0110_2) = 1010$
  - Con esta forma de representar los números, el primer bit coincide con el signo del número (positivos comienzan por 0, negativos comienzan por 1).
  - El rango de números representables con ‘n’ bits va desde -2^{n-1} hasta $2^{n-1}-1$.
  - Existe una única representación del número ‘0’.
  - A la hora de hacer sumas y restas el bit de signo no se trata de forma diferente, sino que se opera con él de forma normal.
  - Las operaciones de suma y resta se hacen igual que si fueran números sin signo (la resta es como la suma, pero complementando el sustraendo).
  - Es la representación utilizada habitualmente en los computadores.

REPRESENTACIÓN DE ENTEROS EN EL COMPUTADOR

- Ejemplos de operaciones ('n' = 4 bits):



Las operaciones se realizan igual independientemente de que los bits representen un número con signo o sin signo.

La única diferencia está en cómo detectar *overflows*.

- En binario puro hay desbordamiento en sumas o restas si se produce acarreo superior.
- En complemento a 2 hay desbordamiento si el signo del resultado es inesperado (el acarreo superior en C2 **siempre** se desprecia).